Computational Statistics: Lab 6

Group 03
Mohsen Pirmoradiyan, Ahmed Alhasan, Yash Pawar

03 March 2020

Question 1: Genetic algorithm

In this assignment, you will try to perform one-dimensional maximization with the help of a genetic algorithm.

1. Define the function

$$f(x) = \frac{x^2}{e^x} - 2\ exp(-(9\ sin\ x)/(x^2 + x + 1))$$

```
RNGversion(min(as.character(getRversion()), "3.6.2"))

f <- function(x){
  (x^2/exp(x)) - 2 * exp(-(9 * sin(x)/(x^2+x+1)))
}
```

2. Define the function crossover(): for two scalars x and y it returns their "kid" as (x+y)=2.

```
crossover <- function(x,y){
  (x+y)/2
}
```

3. Define the function mutate() that for a scalar x returns the result of the integer division x2 mod 30. (Operation mod is denoted in R as %%).

```
mutate <- function(x){
  x^2 %% 30
}
```

4. Write a function that depends on the parameters maxiter and mutprob and:

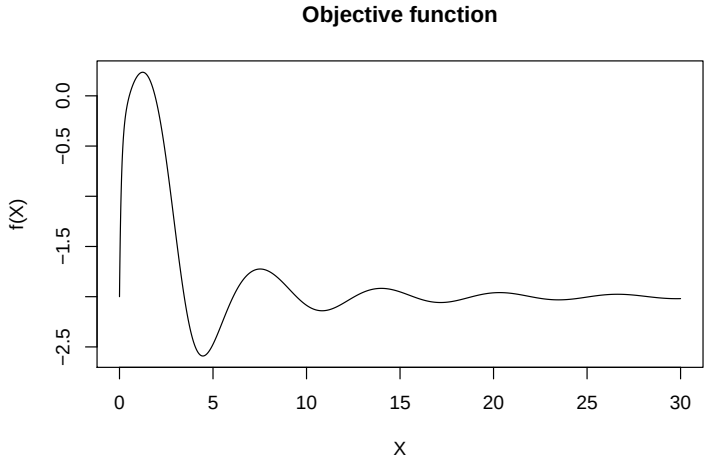
(a) Plots function f in the range from 0 to 30. Do you see any maximum value?
(b) Defines an initial population for the genetic algorithm as X = (0,5,10,15,...,30).
(c) Computes vector Values that contains the function values for each population point.
(d) Performs maxiter iterations where at each iteration

i. Two indexes are randomly sampled from the current population, they are further used as parents (use sample()).
ii. One index with the smallest objective function is selected from the current population, the point is referred to as victim (use order()).

iii. Parents are used to produce a new kid by crossover. Mutate this kid with probability mutprob (use crossover(), mutate()).
iv. The victim is replaced by the kid in the population and the vector Values is updated.
v. The current maximal value of the objective function is saved.

(e) Add the final observations to the current plot in another colour.

```
X <- seq(from = 0, to = 30, by = 0.01)
plot(X, f(X), type = "l", main = "Objective function")
```



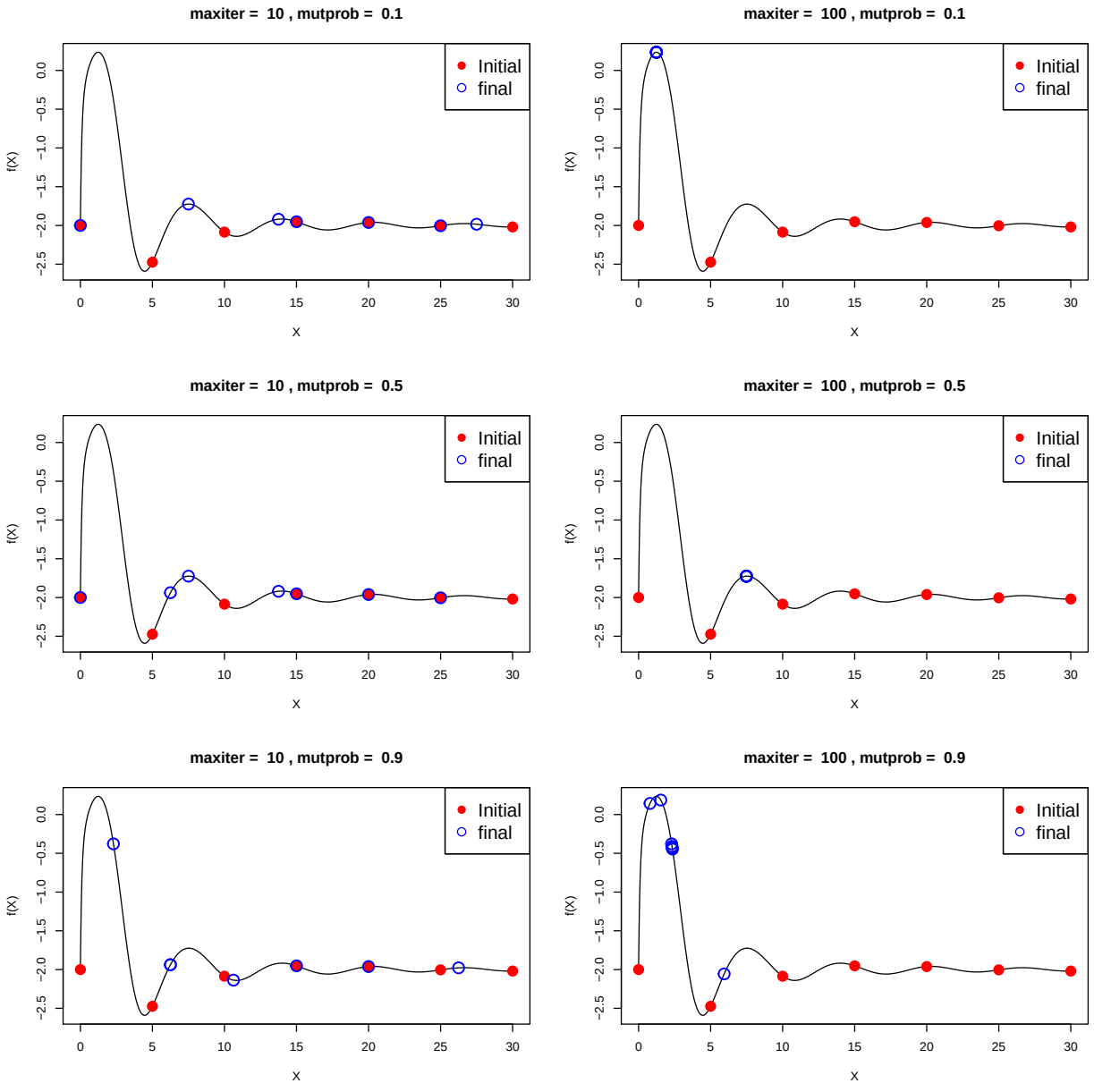

```
genetic <- function(maxiter, mutprob){
  sinusoid  <- seq(from = 0, to = 30, by = 0.01)
  X         <- seq(from = 0, to = 30, by = 5)
  plot(sinusoid, f(sinusoid), type = "l",
       main = paste("maxiter = ", maxiter, ", mutprob = ", mutprob),
       xlab = "X",
       ylab = "f(X)")
  points(X, f(X), col = "red", pch = 16, cex = 2)

  Values  <- f(X)
  max_val <- -Inf
  for(i in 1:maxiter){
    parents <- sample(X, size = 2)
    victim  <- order(Values)[1]
    kid <- crossover(x = parents[1], y = parents[2])
    kid <- ifelse(mutprob > runif(1), mutate(kid), kid)
    X[victim] <- kid
    Values    <- f(X)
    max_val   <- max(max_val, max(Values))
  }
  list(optimum = max_val, population = X, Values = Values)

  points(X, Values, pch = 21, col = "blue", cex = 2, lwd = 1.5)
  legend(x = "topright", legend = c("Initial", "final"), pch = c(16,21),
         col = c("red", "blue"), cex = c(1.5, 1.5))
```

```
}

maxiter = c(10, 100)
mutprob = c(0.1, 0.5, 0.9)
par(mfrow = c(3,2))
for(prob in mutprob){
  for(iter in maxiter){
    set.seed(12345)
    genetic(iter, prob)
  }
}
```



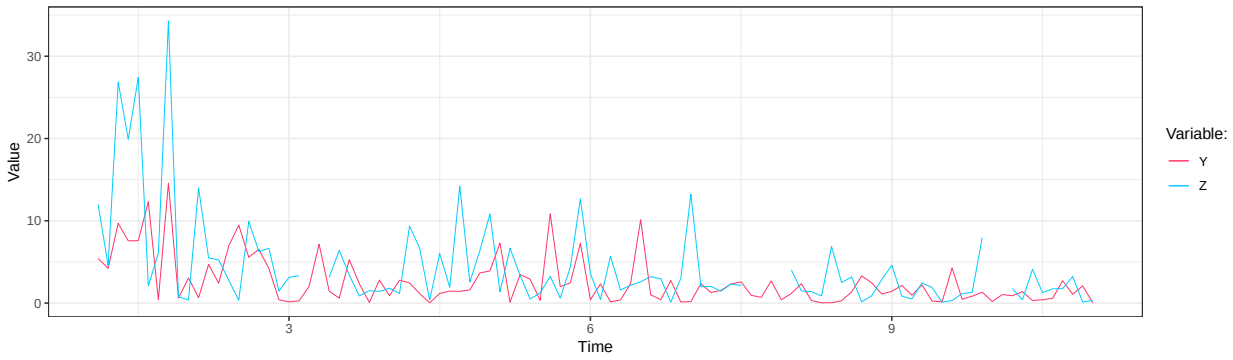
- The initial population is well diversed over the inspected interval and that will "ensure that the solution space is adequately searched, especially in the earlier stages of the optimization process"(Deepti Gupta 2012)
- In this case fewer iterations are not enough to converge to global maximum and since more iteration is need to fully search the solution space
- The crossover is applied to all the population to generate new offsprings by taking the mean of the parents, in part this will increase the diversity since the kids will be at the most distance from the parents.
  And on the other part the mean function could produce new generation identical to the old one when the middle value is another old generation and given the low starting population this will have great effect unless we provide a condition to deselect any old generation result from the crossover
- The mutation provide exploration and prevent the algorithm from converging to local maximum, small mutation rate GA could cause premature convergence and could stuck anywhere in the fitness function, however because the fitness function is not so complex GA converged easily to global maximum with mutation rate = 0.1.
- In the mean while higher mutation rates provide more exploration and that could cause GA to not settle long time in the global maximum because there is high probability we are replacing high fit parents with new generation that is not necessarily better fit

Question 2: EM algorithm

The data file physical.csv describes a behavior of two related physical processes Y = Y (X) and Z = Z(X).

1. Make a time series plot describing dependence of Z and Y versus X. Does it seem that two processes are related to each other? What can you say about the variation of the response values with respect to X?

```
physical <- read.csv("../Data/physical1.csv")
library(ggplot2)
ggplot(physical, aes(x = X)) +
  geom_line(aes(y = Y, col = "Y"), size = .3) +
  geom_line(aes(y = Z, col = "Z"), size = .3) +
  scale_color_manual(values = c("#FF3366", "#00CCFF"), name = "Variable: ") +
  labs(y = "Value", x = "Time") +
  theme_bw()
```



- Variance of Y and Z is decreasing while X increases

2. Note that there are some missing values of Z in the data which implies problems in estimating models by maximum likelihood. Use the following model

$$Y_i \sim exp(X_i/\lambda), \quad Z_i \sim exp(X_i/2\lambda)$$

where λ is some unknown parameter.
The goal is to derive an EM algorithm that estimates λ.

$$p(Y_i|\lambda) = \frac{X_i}{\lambda} \exp(-\frac{X_i Y_i}{\lambda}), \quad p(Z_i|\lambda) = \frac{X_i}{2\lambda} \exp(-\frac{X_i Z_i}{2\lambda})$$

Since there is little correlation between Y and Z we can assume they are independent so we can get joint distribution by only multiplying them together and derive the likelihood accordingly

$$L(\lambda|Y_i, Z_i) = \prod_{i=1}^{n} p(Y_i|\lambda).p(Z_i|\lambda) = \prod_{i=1}^{n} \left(\frac{X_i}{\lambda} \exp(-\frac{X_i Y_i}{\lambda})\right)\left(\frac{X_i}{2\lambda} \exp(-\frac{X_i Z_i}{2\lambda})\right)$$

$$= \frac{\prod_{i=1}^{n} X_i^2}{(2\lambda^2)^n} \exp\Big(-\frac{1}{\lambda}\sum_{i=1}^{n} X_i Y_i - \frac{1}{2\lambda}\sum_{i=1}^{n} X_i Z_i \Big)$$

By taking the log-liklihood we get:

$$l(\lambda|Y_i, Z_i) = 2\ln\Big(\prod_{i=1}^{n} X_i\Big) - \Big(n\ln(2) + 2n\ln(\lambda)\Big) - \frac{1}{\lambda}\sum_{i=1}^{n} X_i Y_i - \frac{1}{2\lambda}\sum_{i=1}^{n} X_i Z_i$$

Because Z has missing values we need to seperate the last term into two parts

$$l(\lambda|Y_i, Z_i) = 2\ln\Big(\prod_{i=1}^{n} X_i\Big) - \Big(n\ln(2) + 2n\ln(\lambda)\Big) - \frac{1}{\lambda}\sum_{i=1}^{n} X_i Y_i - \frac{1}{2\lambda}\sum_{i\in Observed} X_i Z_i - \frac{1}{2\lambda}\sum_{j\in Missing} X_j Z_j$$

We can estimate the missing values by mutating them with the estimated value of λ given Y and Z

$$E[l(\lambda|Y_i, Z_i)] = E\left[2\ln\Big(\prod_{i=1}^{n} X_i\Big) - \Big(n\ln(2) + 2n\ln(\lambda)\Big) - \frac{1}{\lambda}\sum_{i=1}^{n} X_i Y_i - \frac{1}{2\lambda}\sum_{i\in Observed} X_i Z_i - \frac{1}{2\lambda}\sum_{j\in Missing} X_j Z_j\right]$$

For the exponential distribution the expected value of Z would be

$$E[Z_j] = \frac{2\lambda_t}{X_j}, \quad \textit{where } \lambda_t \textit{ is the lambda from last iteration}$$

$$E\left[\frac{1}{2\lambda}\sum_{j\in Missing} X_j Z_j\right] = M\frac{\lambda_t}{\lambda}, \quad \textit{where } M \textit{ is the number of missing values}$$

Now we have a log-likelihood function that can be maximized

$$E[l(\lambda|Y_i, Z_i)] = 2\ln\Big(\prod_{i=1}^{n} X_i\Big) - \Big(n\ln(2) + 2n\ln(\lambda)\Big) - \frac{1}{\lambda}\sum_{i=1}^{n} X_i Y_i - \frac{1}{2\lambda}\sum_{i\in K} X_i Z_i - M\frac{\lambda_t}{\lambda}$$

In the M step we take the partial derivative with respect to λ:

$$\frac{\partial E[l(\lambda|Y_i, Z_i)]}{\partial\lambda} = -\frac{2n}{\lambda} + \frac{1}{\lambda^2}\sum_{i=1}^{n} X_i Y_i + \frac{1}{2\lambda^2}\sum_{i\in Observed} X_i Z_i + M\frac{\lambda_t}{\lambda^2} = 0$$

$$\lambda_{t+1} = \frac{1}{2n}\left(\sum_{i=1}^{n} X_i Y_i + \frac{1}{2}\sum_{i\in Observed} X_i Z_i + M\lambda_t\right)$$

Where λ_{t+1} is the next λ

3. Implement this algorithm in R, use $\lambda_0 = 100$ and convergence criterion "stop if the change in λ is less than 0.001". What is the optimal λ and how many iterations were required to compute it?

```
n   <- dim(physical)[1]
obs <- which(!is.na(physical$Z))
mis <- which(is.na(physical$Z))
M   <- length(mis)

dif    <- Inf
lambda <- 100
t      <- 0
repeat{
  lambda_new <- (sum(physical$X * physical$Y) +
                 M * lambda +
                 0.5 * sum(physical$X[obs] * physical$Z[obs])) / (2*n)

  dif    <- abs(lambda_new - lambda)
  lambda <- lambda_new
  t      <- t + 1

  print(lambda)
  if(dif < 0.001)break
}
```

```
## [1] 14.26782
## [1] 10.83853
## [1] 10.70136
## [1] 10.69587
## [1] 10.69566
```

```
cat("\nOptimal Lambda:", lambda)
```

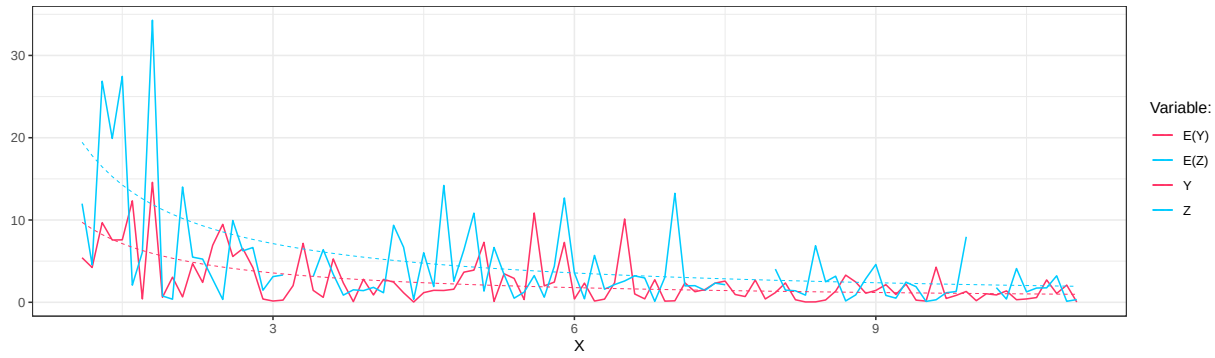
```
##
## Optimal Lambda: 10.69566
```

```
cat("\nNumber of iterations:", t)
```

```
##
## Number of iterations: 5
```

4. Plot E [Y] and E[Z] versus X in the same plot as Y and Z versus X. Comment whether the computed λ seems to be reasonable.

```
library(ggplot2)
ggplot(physical, aes(x = X))+
  geom_line(mapping = aes(y = Y, color="Y")) +
  geom_line(mapping = aes(y = Z, color="Z")) +
  geom_line(mapping = aes(y = lambda/X, color="E(Y)"), linetype = "dashed", size = .3) +
  geom_line(mapping = aes(y = 2*lambda/X, color="E(Z)"), linetype = "dashed", size = .3) +
  scale_color_manual(values = c("#FF3366", "#00CCFF", "#FF3366", "#00CCFF"), name = "Variable: ") +
  ylab("") +
  theme(legend.position = "bottom") +
  theme_bw()
```


30
20
10
0
3
6
9
X
Variable:
E(Y)
E(Z)
Y
Z

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
RNGversion(min(as.character(getRversion()), "3.6.2"))

f <- function(x){
  (x^2/exp(x)) - 2 * exp(-(9 * sin(x)/(x^2+x+1)))
}
crossover <- function(x,y){
  (x+y)/2
}
mutate <- function(x){
  x^2 %% 30
}
X <- seq(from = 0, to = 30, by = 0.01)
plot(X, f(X), type = "l", main = "Objective function")
genetic <- function(maxiter, mutprob){
  sinusoid  <- seq(from = 0, to = 30, by = 0.01)
  X         <- seq(from = 0, to = 30, by = 5)
  plot(sinusoid, f(sinusoid), type = "l",
       main = paste("maxiter = ", maxiter, ", mutprob = ", mutprob),
       xlab = "X",
       ylab = "f(X)")
  points(X, f(X), col = "red", pch = 16, cex = 2)

  Values  <- f(X)
  max_val <- -Inf
  for(i in 1:maxiter){
    parents <- sample(X, size = 2)
    victim  <- order(Values)[1]
    kid <- crossover(x = parents[1], y = parents[2])
    kid <- ifelse(mutprob > runif(1), mutate(kid), kid)
    X[victim] <- kid
    Values    <- f(X)
    max_val   <- max(max_val, max(Values))
  }
  list(optimum = max_val, population = X, Values = Values)

  points(X, Values, pch = 21, col = "blue", cex = 2, lwd = 1.5)
  legend(x = "topright", legend = c("Initial", "final"), pch = c(16,21),
         col = c("red", "blue"), cex = c(1.5, 1.5))
}

maxiter = c(10, 100)
mutprob = c(0.1, 0.5, 0.9)
par(mfrow = c(3,2))
for(prob in mutprob){
  for(iter in maxiter){
    set.seed(12345)
    genetic(iter, prob)
  }
}
physical <- read.csv("../Data/physical1.csv")
```

```r
library(ggplot2)
ggplot(physical, aes(x = X)) +
  geom_line(aes(y = Y, col = "Y"), size = .3) +
  geom_line(aes(y = Z, col = "Z"), size = .3) +
  scale_color_manual(values = c("#FF3366", "#00CCFF"), name = "Variable: ") +
  labs(y = "Value", x = "Time") +
  theme_bw()
n   <- dim(physical)[1]
obs <- which(!is.na(physical$Z))
mis <- which(is.na(physical$Z))
M   <- length(mis)

dif    <- Inf
lambda <- 100
t      <- 0
repeat{
  lambda_new <- (sum(physical$X * physical$Y) +
                 M * lambda +
                 0.5 * sum(physical$X[obs] * physical$Z[obs])) / (2*n)

  dif    <- abs(lambda_new - lambda)
  lambda <- lambda_new
  t      <- t + 1

  print(lambda)
  if(dif < 0.001)break
}
cat("\nOptimal Lambda:", lambda)
cat("\nNumber of iterations:", t)
library(ggplot2)
ggplot(physical, aes(x = X))+
  geom_line(mapping = aes(y = Y, color="Y")) +
  geom_line(mapping = aes(y = Z, color="Z")) +
  geom_line(mapping = aes(y = lambda/X, color="E(Y)"), linetype = "dashed", size = .3) +
  geom_line(mapping = aes(y = 2*lambda/X, color="E(Z)"), linetype = "dashed", size = .3) +
  scale_color_manual(values = c("#FF3366", "#00CCFF", "#FF3366", "#00CCFF"), name = "Variable: ") +
  ylab("") +
  theme(legend.position = "bottom") +
  theme_bw()
```

References

Deepti Gupta, Shabina Ghafir. 2012. "An Overview of Methods Maintaining Diversity in Genetic Algorithms." *International Journal of Emerging Technology and Advanced Engineering* 2.